Non-supervised Learning & Decision Making

Danilo J. Rezende

Hammers & Nails 2019, Weizmann Institute of Science, Israel
"The revolution will not be supervised"  (Yann Lecun, 2017)
Topics

- General discussion about Reinforcement Learning
- Why build models?
- Using generative models for representations learning:
  - GQN
  - SimCore
- Some thoughts on applications of group theory in ML
Reinforcement Learning
Let's consider a discrete Markov Decision Process (MDP)
RL in a discrete world

Set of States
Set of Rewards
Set of Actions
Reward function
Model/Environment

\[ S = \{1, \ldots, N_s\} \]
\[ \mathcal{R} = \{1, \ldots, N_r\} \]
\[ \mathcal{A} = \{1, \ldots, N_a\} \]
\[ r : S \times \mathcal{A} \rightarrow \mathcal{R} \]
\[ m : S \times \mathcal{A} \rightarrow S \]
Let's consider a discrete world MDP

\[
Q(s_0, a_0) = \max_{\text{all possible paths}} \sum_{t=1}^{H} \gamma^t r(s_t, a_t)
\]

Optimal decision

\[
a^*(s_0) = \arg\max_a Q(s_0, a)
\]
Let's consider a discrete world MDP

\[ Q(s_0, a_0) = \max_{\text{all possible paths}} \sum_{t=1}^{H} \gamma^t r(s_t, a_t) \]

- Set of States: \( S = \{1, \ldots, N_s\} \)
- Set of Rewards: \( \mathcal{R} = \{1, \ldots, N_r\} \)
- Set of Actions: \( \mathcal{A} = \{1, \ldots, N_a\} \)
- Reward function: \( r : S \times A \rightarrow \mathcal{R} \)
- Model/Environment: \( m : S \times A \rightarrow S \)
- Q-function: \( Q : S \times A \rightarrow \mathcal{R}^H \)
How much data do we need to learn $r$, $m$ and $Q$?

Amount of Data to learn $f \approx e^{\text{Description Length of } f}$

\[
\begin{align*}
\text{DL}(r) &= N_s N_a \log_2(N_r) \\
\text{DL}(m) &= N_s N_a \log_2(N_s) \\
\text{DL}(Q) &= N_s N_a H \log_2(N_r)
\end{align*}
\]
When is it worth learning a model?

\[ DL(r + m) = N_s N_a \log_2(N_r) + N_s N_a \log_2(N_s) \]
\[ DL(Q) = N_s N_a H \log_2(N_r) \]

Hypothesis: It will be favorable to learn a model when
When is it worth learning a model?

\[ DL(r + m) = N_s N_a \log_2(N_r) + N_s N_a \log_2(N_s) \]
\[ DL(Q) = N_s N_a H \log_2(N_r) \]

Hypothesis: It will be favorable to learn a model when

\[ DL(Q) > DL(r + m) \]

\[ (H - 1) \log_2(N_r) > \log_2(N_s) \]
When is it worth learning a model?

$$(H - 1) \log_2(N_r) > \log_2(N_s)$$

In general, we operate in a regime where $N_s \gg N_r, H$. So it seems that we would always prefer to learn $Q$ from scratch rather than learning a model first.
When is it worth learning a model? Many tasks, a single world

\[ L(H - 1) \log_2(N_r) > \log_2(N_s) \]

For a sufficiently large number of different tasks \( L \), it will require less data if we learn a model first.
Summary

● Learning a model first and using that to compute Q-functions via Monte-Carlo can be more data-efficient in some cases, specially in multi-task problems.

● Models offer other advantages in addition to planning:
  ○ Notion of uncertainty or novelty
  ○ Unsupervised learning of useful features
  ○ Unsupervised learning to use complex memory architectures
RL in a stochastic and partially observed world
RL in a stochastic and partially observed world
RL in a stochastic world

\[ \pi(a_t | x_0, \ldots, t, a_0, \ldots, (t-1)) \]

\[ m(x_t | x_0, \ldots, (t-1), a_0, \ldots, (t)) \]

\[ \tau = \{(x_1, a_1), \ldots, (x_t, a_t), \ldots (x_H, a_H)\} \]

\[ Q^\pi(s_0, a_0) = \mathbb{E}_{p(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \middle| s_0, do(a_0) \right] \]
RL in a stochastic world

\[
\pi(a_t | x_0, ..., t, a_0, ..., (t-1))
\]

\[
m(x_t | x_0, ..., (t-1), a_0, ..., (t))
\]

\[
\tau = \{(x_1, a_1), \ldots, (x_t, a_t), \ldots (x_H, a_H)\}
\]

\[
Q^\pi(s_0, a_0) = \mathbb{E}_{p(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \middle| s_0, \text{do}(a_0) \right]
\]
RL in a stochastic world

Stochastic Simulator/Model

Feynman-Kac lemma

State distributions (Fokker-Planck)

Control via path-integrals

Path Integral Formulation of Stochastic Optimal Control with Generalized Costs Yang et al
A Generalized Path Integral Control Approach to Reinforcement Learning Theodorou et al
RL in a stochastic world

\[ Q^\pi(s_0, a_0) = \mathbb{E}_{p(\tau)} \left[ \sum_{t} \gamma^t r(s_t, a_t) \bigg| s_0, \text{do}(a_0) \right] \]
RL in a stochastic world

\[ Q^\pi(s_0, a_0) = \mathbb{E}_{p(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \mid s_0, \text{do}(a_0) \right] \]

\[ Q^\pi(s_0, a_0) \approx \frac{1}{N} \sum_k \sum_t \gamma^t r(s^k_t, a^k_t) \]
RL in a complex world

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures
Lasse Espeholt, Hubert Soyer
RL in a stochastic world

$$Q^\pi(s_0, a_0) = \mathbb{E}_{p(\tau)} \left[ \sum_t \gamma^t r(s_t, a_t) \bigg| s_0, \text{do}(a_0) \right]$$

$$Q^\pi(s_0, a_0) \approx \frac{1}{N} \sum_k \sum_t \gamma^t r(s_t^k, a_t^k)$$

$$Q^\pi(s_0, a_0) \approx \frac{1}{N} \sum_k \sum_t \gamma^t r(s_t^k, a_t^k) \frac{\pi_{\text{current}}(a_t^k)}{\pi_{\text{old}}(a_t^k)}$$

**IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures**
Lasse Espeholt, Hubert Soyer
"Canonical" Agent

IMPALA: Scalable Distributed Deep-RL with Importance Weighted Actor-Learner Architectures
Lasse Espeholt, Hubert Soyer
Models and Reinforcement Learning
Neural Scene Representation and Rendering


DeepMind
The Model

[Diagram of a neural scene representation and rendering model]

The Model

The Model

The Model

Near-Deterministic Predictions

Data fusion and predicting with uncertainty

Quantifying uncertainty

Quantifying uncertainty

Quantifying uncertainty

Viewpoint invariance of the learned scene representations

Factorization of the learned scene representations

Factorization of the learned scene representations

Learning useful representations for control

Deep-Learning with Uncertainty


Slide credit: Marta Garnelo
• GQN can learn factored scene representations
• It is also a kind of meta-learning model (learning to do one-shot scene inference)
• It doesn't have a notion of time
• It requires knowledge about "camera locations"
Shaping Belief States with Generative Environment Models for RL


DeepMind
Example (actual result)

What agent sees

Actual top down view

Top down view reconstructed from agent rep.

Without belief state training

With belief state training

Level = rat_goal_driven

Score

Data seen
Model Rollouts
Formation of belief states
Top down view reconstruction error (mapping)

1) Longer simulations improve map decoding performance (if proper generative model is used)
2) Generative model works better than deterministic decoder and CPC
3) Kanerva Machine episodic memory works the best
Position and orientation prediction (localization)

1) Longer simulations improve localization
2) CPC works the best
More complex naturalistic environment
RL Performance

**DM Lab**

- Level = rat_goal_driven
- Level = rat_goal_doors
- Level = rat_goal_driven_large
- Level = keys_doors_random

**Voxel Lab**

- Level = Food
- Level = HighFood
- Level = Cliff
- Level = BridgeFood

Graphs show performance metrics such as score over data seen. Different agents are evaluated, including LSTM, RMA + SimCore, RMA, and LSTM.
Learns to build stairs and towers
Learns to imagine building them

Building stairs

Building towers

input rollout
Naturalistic landscape - imagines eating the yellow block
Towards a Definition of Disentangled Representations

Irina Higgins, David Amos, David Pfau, Sebastien Racaniere, Loic Matthey, Danilo Rezende, Alexander Lerchner
Motivations

- What is the role Group theory in "feature/representation learning"?
- invariance vs covariance vs equivariance
- When is "Disentanglement" useful?
- Our intuition should not rely on a specific coordinate system
- Lack of universality => lack of generalisation to domains where we have no intuition
A group $G$ is a set endowed with a binary operator $o$. The operator must satisfy the following properties:

- **Closure**: $x_1, x_2 \in G$, then $x_1 \circ x_2$ is in $G$
- **Identity**: there is an $e \in G$ such that $e \circ x = x$ (for all $x \in G$)
- **Inverse**: for any $x \in G$ there is a $y$ such that $y \circ x = e$
- **Associativity**: $x_1 \circ (x_2 \circ x_3) = (x_1 \circ x_2) \circ x_3$

**Example:**

$G = (\text{Reals}, o=+, e = 0)$

**Example:**

$G = (\text{Rotation matrices}, o=\text{matrix product}, e = I)$
Group theory 1:1
What is a representation of a group?

A representation of a group is a mapping from group element $g$ to an operator $\rho(g)$ acting on a different space $H$ with operator $x$. Such that $\rho(g_1 \circ g_2) = \rho(g_1) \times \rho(g_2)$.
What is an irreducible representation of a group?

Representations that leave invariant some subset of H that cannot be broken down into smaller invariant subsets.
A Lie Group is a group that is also a manifold (e.g. Translations and Rotations)
We say that our features $f$ are *invariant* under a symmetry group $G$ iff $f(g \circ \text{data}) = f(\text{data})$ for any $g$ in $G$. Example: output of pooling layers.

We say that we have *equivariant features* $f$ under a symmetry group $G$ if $f(g \circ \text{data}) = J \times f(\text{data})$. Example: output of convolution layers, vector fields.
Learning the Irreducible Representations of Commutative Lie Groups

2D/3D Rotation-Invariant Detection using Equivariant Filters and Kernel Weighted Mapping
Invariance, Equivariance, Classifiers and Generative Models

When building classifiers it is desirable that its output to be invariant under some groups (e.g. translation, rotation).
That is, we want to build both features and losses such that $\text{Loss}(g \circ \text{data}) = \text{Loss}(\text{data})$, where $\circ = 2\text{D Translations} \times 2\text{D Rotations}$

However, when building generative models, we should seek invariant losses, but the representations should be equivariant. This is because invariant representations loose information, destroying the ability to reconstruct the data.
General Recipe to build invariant networks

First, build equivariant features

\[ \phi(g \circ x) = T_g \circ \phi(x) \]

Second, apply a ‘pooling’ operator

\[ \psi(x) = \int d\mu(g)T_g \circ \phi(x) \]

Example: convnets
The elementary components of a system are the irreducible representations of the symmetry group of the system. Example: entire physics.
Why does it matter to AI?

3D OBJECTS are irreducible representations of 3D Translation x Rotation x Scale groups (Galileo Group)
Weyl principle $\Rightarrow$ Irreducible representations $\Leftrightarrow$ Disentanglement (since we have broken down our representations into the smallest possible invariant sets)
Towards a Definition of Disentangled Representations

A space $Z$ is **disentangled** with respect to a group decomposition $G = G_1 \times G_2 \times \ldots \times G_N$ if:

1) There is an action $\cdot : G \times Z \rightarrow Z$

2) The map $f : W \rightarrow Z$ is equivariant between the actions on $W$ and $Z$

3) There is a decomposition $Z = Z_1 \times Z_2 \times \ldots \times Z_N$ such that each $Z_i$ is fixed by the action of all $G_j, j \neq i$ and affected only by $G_i$

The actions are assumed to preserve any structure of $Z$ (e.g. be linear or continuous).
Towards a Definition of Disentangled Representations
Summary

- The language of symmetry groups and differential geometry allows for generalisation of useful tools such as conv operators beyond classification of images.

- Such ideas have only been superficially explored in ML.

- Do you want objects to emerge from neural nets + data? Let's build "Weyl machines" (or automated physicists).
Thanks to great collaborators

Karol Gregor
Marta Garnelo
Ali Eslami
Frederic Besse
Fabio Viola
Hamza Merzic
Yan Wu
Theophane Weber

Daniel Zoran
Alex Mott
Irina Higgins
Sebastian Racaniere
David Pfau
Mihaela Rosca
Marco Fraccaro
Aaron Van der Oord
Daan Wierstra